CS 726: Homework #5

Posted: 03/30/2020, due: 04/13/2020 by 5pm on Canvas

Please typeset or write your solutions neatly! If we cannot read it, we cannot grade it.

Note: You can use the results we have proved in class – no need to prove them again. To implement Euclidean projections onto simplex, you can use the provided `ProjectOntoSimplex.m` and `ProjectOntoEllOneBall.m` files (available under 'Files' on Canvas). If you are coding in Python, you can find similar implementations online (for example, take a look at https://gist.github.com/daien/1272551/edd95a6154106f8e28209a1c7964623ef8397246). The same rules as in previous homework assignments regarding the use of Python for coding apply.
Your code needs to compile without any errors to receive any points. You need to justify all your answers.

Q 1 (The importance of choosing the appropriate geometry)**.** In this question, you are asked to implement Projected Gradient Descent and Mirror Descent to solve the following problem:

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}), \tag{P}$$

where $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \geq \mathbf{0},\ \mathbf{1}^T\mathbf{x} = 1\}$ is the probability simplex and f is defined by:

$$f(\mathbf{x}) = \max_{1\leq i\leq n} \left\langle \mathbf{z}^i, \mathbf{x} \right\rangle, \tag{1}$$

where $\mathbf{z}^i$ are independent Rademacher vectors. Namely, $\forall i, j \in \{1, \ldots, n\}$, $z^i_j = +1$ with probability $\frac{1}{2}$ and $z^i_j = -1$ with probability $\frac{1}{2}$, independently over $i, j \in \{1, \ldots, n\}$.

Is (P) a convex optimization problem? Justify your answer.

What is the Lipschitz constant M_2 of f with respect to $\|\cdot\|_2$? What is the Lipschitz constant M_1 of f with respect to $\|\cdot\|_1$? (We will accept answers that just bound one of the subgradients at each of the possible query points, in the appropriate norm.) What is $D_2 = \max_{\mathbf{x},\mathbf{y}\in\mathcal{X}} \|\mathbf{x} - \mathbf{y}\|_2$? What is $D_1 = \max_{\mathbf{x},\mathbf{y}\in\mathcal{X}} \|\mathbf{x} - \mathbf{y}\|_1$?

The variant of Mirror Descent you should implement will use the negative entropy regularizer $\psi(\mathbf{x}) = \sum_{i=1}^n x_i \log(x_i)$ (see the previous homework assignment for the definition of Mirror Descent). Note that this $\psi(\mathbf{x})$ is 1-strongly convex with respect to the ℓ_1 norm (you don't need to prove this). Note also that the iterates of the Mirror Descent algorithm can be written in closed form for this choice of ψ. Write the formula for the iterates $\mathbf{x}_{k+1}$ of Mirror Descent explicitly as a function of the step size a_k, previous iterate $\mathbf{x}_k$, and a subgradient $\mathbf{g}_{\mathbf{x}_k}$ of f at $\mathbf{x}_k$. Prove that the formula for $\mathbf{x}_{k+1}$ is invariant to translation of $a_k\mathbf{g}_{\mathbf{x}_k}$, meaning that it gives the same result for $a_k\mathbf{g}_{\mathbf{x}_k}$ and $a_k\mathbf{g}_{\mathbf{x}_k} + C\mathbf{1}$, for any constant $C \in \mathbb{R}$. This is important for numerical stability of your algorithm – for the algorithm not to encounter numerical issues, you will want to compute $\mathbf{x}_{k+1}$ using $a_k\mathbf{g}_{\mathbf{x}_k} + C\mathbf{1}$, with $C = -a_k \min_{1\leq i\leq n}(\mathbf{g}_{\mathbf{x}_k})_i$.

You should initialize both methods at $\mathbf{x}_0 = \frac{1}{n}\mathbf{1}$. What is $\max_{\mathbf{u}\in\mathcal{X}} D_\psi(\mathbf{u}, \mathbf{x}_0)$?

To implement the methods, you should take $n = 500$ and run them for $K = 2000$ iterations. Your step sizes should be $a_i = 1/(M\sqrt{i})$, where $M \in \{M_1, M_2\}$ is the appropriate Lipschitz constant for each of the two methods. Your code should output two plots, comparing the function values of the two methods aginst the iteration count. The first plot should use function evaluations at individual iterates/query points of the algorithms $\mathbf{x}_i$. The second plot should use function evaluations at the algorithm's output points $\mathbf{x}^{\text{out}}_i = \frac{1}{A_i}\sum_{j=0}^i a_j\mathbf{x}_j$. Discuss your results. What can you conclude from the first plot? Now observe the second plot. Which algorithm is faster? Use the theoretical results we derived in the last homework and in class to explain why.

Q 2 (Comparing Frank-Wolfe and PGD)**.** In this question, you are asked to compare the Frank-Wolfe method we saw in class to Projected Gradient Descent, on the following problem instance:

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}),$$

where $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_1 \leq 1\}$ is the ℓ_1 ball and $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{B}\mathbf{x} - \mathbf{b}\|_2^2$ is a quadratic function defined by: (i) $\mathbf{B}$, an $p \times n$ matrix whose entries are i.i.d. Gaussian with mean zero and variance one, where $n = 200$ and $p = 10$; and (ii) $\mathbf{b} \in \mathbb{R}^p$ is the vector of all 1's.

What are the vertices of $\mathcal{X}$? How do you compute $\mathbf{v}_k = \operatorname{argmin}_{\mathbf{u}\in\mathcal{X}}\{\langle \nabla f(\mathbf{x}_k), \mathbf{u} - \mathbf{x}_k\rangle\}$? Write it in closed form. For PGD, you can use the provided code to implement projections onto $\mathcal{X}$.

Run both algorithms 30 times for 300 iterations. You should initialize both algorithms at $\mathbf{x}_0 = \mathbf{e}_1$, where $\mathbf{e}_1$ is the first basis vector. Your code should output a plot that shows $\frac{f(\mathbf{x}_k)}{f(\mathbf{x}_0)}$ against the iteration count for PGD and the Frank-Wolfe algorithm *averaged over 30 independent runs*. Discuss the results. Do you see what you expect from the theoretical results we derived in class?

You should also consider, for each of the algorithms, the solution points at the first iteration for which $\frac{f(\mathbf{x}_k)}{f(\mathbf{x}_0)} \leq 10^{-1}$. Your code should output the average sparsity (number of nonzero elements) of these points for both algorithms, over 30 independent runs. Which of the two algorithms constructs sparser solutions (solutions with fewer nonzero elements)? Can you explain why?

Q 3 (First-order methods in the presence of gradient noise)**.** In this question, you should implement three algorithms: (i) Nesterov's AGD, (ii) standard gradient descent with step size $1/L$, and (iii) (projected) stochastic gradient descent with averaging (from class) with the step size $a_k = 1/(L\sqrt{k})$. Your algorithms should **not** work with the exact gradients; instead, you should implement them with "noisy" gradient oracles $\mathbf{g}(\mathbf{x}, \boldsymbol{\xi}) = \nabla f(\mathbf{x}) + \varepsilon\boldsymbol{\xi}$, where $\boldsymbol{\xi}$ are vectors with i.i.d. mean-zero variance-one Gaussian elements, and ε is an algorithm parameter.

The problem instance is $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{M}\mathbf{x} - \mathbf{b}^T\mathbf{x}$, where $\mathbf{M} \in \mathbb{R}^{n\times n}$ is a tridiagonal matrix with 2's on the main diagonal and -1's on the remaining two diagonals (as in previous problem sets) and $\mathbf{b} = \mathbf{e}_1$.

Your algorithms should all be initialized at $\mathbf{x} = \mathbf{0}$.

Your code should output three plots, produced for three different values of ε : (i) $\varepsilon = 10^{-5}$, (ii) $\varepsilon = 10^{-3}$, and (iii) $\varepsilon = 10^{-1}$. In all the plots, $n = 200$. Each plot should show the optimality gap (on a log scale) against the iteration count, for all three algorithms.

Discuss your results. What do you observe? Which algorithm is fastest? Which one is the most stable? Which one would you use in practice, under what circumstances, and why?